11p.

On the effective definition of "Random Sequence"

Michael Levin
Marvin Minsky
Roland Silver

Revised version of AI Memo 36, RLE and MIT Computation Center

This empty page was substituted for a blank page in the original document.

The Problem of the Effective Definition of "random sequence"

Mathematicians have always had difficulty in coming to agreement over what is meant by "randomness". In order to agree on a formal model for a "random process", we have to agree on what intuitive aspects of the matter we want to build into our system. The most prominent point of agreement is that the process should be unpredictable, but this is in itself a very small beginning. The solution that has become conventional in modern mathematics is based on the notion of "random variable", a highly technical notion in which the basic process is represented as a certain kind of infinite function-space. This space contains all possible observed behavior sequences together with a "measure" structure which enables one to calculate the relative frequency of certain ("measurable") complex events. "Event" here usually refers to a whole class of behaviors.

In this generally accepted theory, the individual "random sequence" plays a very small role, because its individual measure or probability is zero, in non-trivial cases. Thus, if one flips a fair coin, the probability of any particular sequence is evidently zero. Furthermore, for this situation all individual sequences are equally likely. Yet the particular sequence is "obviously" non-random! If we can say that a particular sequence is not random, shouldn't we be able to say that some other sequence is random? Until one becomes ever so sophisticated, this feeling lingers. But how could one characterize a single random sequence? One would have to be

careful, for if the description is too specific, the sequence might lose the precious quality of controlled unpredictability.

It would be very pleasing and useful, if one could set up a satisfactory theory in which it was meaningful to talk of an individual random sequence.* It would be even more pleasing and useful if one could define this in some effective manner.

It seems safe to say that we could not admit any computable sequences to the category of random sequences. For a computable sequence has the character that one needs only a finite amount of information in order to be able to make perfect predictions.**

On the other hand, it might seem safe to say that a sequence should be considered random if there is no way to make a "better than chance" prediction about its next term, given the past behavior.

The constraint that there be "no way to make a 'better than chance' prediction" has the right spirit, but it is far too vague to be useful as a mathematical condition.

We will formalize the notion is a certain "effective" way and then show that in at least that sense, there will exist computable sequences which satisfy our definition of randomness. This result may be viewed in two ways. On the one hand it shows the futility of trying to make effective such a definition, if one's goal is to obtain an absolute notion of randomness. On the other hand, it does show that there are computable sequences so irregular that one can be constructed to defeat any such systematic attempt: that is, there are arbitrarily "disordered" yet effectively-defined "pseudo-random" sequences.

** Indeed, the reader unfamiliar with the definition of computability can take this as the definition.

* The name of R. Von Mises is associated with such attempts. See ref. [4].

We will use a very crude measure of prediction ability. Let M be a (Turing) machine which computes some binary function of finite binary sequences. That is, given any finite string of 0's and 1's, M will produce a 0 or a 1. Given some infinite sequence S(1), S(2), . . . we can have M produce, for each term S(t), a digit M(t+1) which we will interpret as a"guess" about what will be the next term S(t+1) of the sequence S. We keep a "cumulative score" G(t) for M , adding 1 to it for each correct gness and subtracting 1 for each incorrect guess. If M can consistently do " better than chance" then we might find, in the long run, that

$$\lim_{t\to\infty} \frac{G(t)}{t} = \alpha > 0$$

(assuming that the limit exists at all).

Clearly, for any particular machine M we could find computable sequences S(t) which would i) force the limit to 0, or ii) force the limit to 1 unity, and iii) prevent the relative score from converging to any limit at all. On the other hand, given first any computable sequence we could find a machine M which perfectly predicts the sequence (by using the same machine that generates the sequence). The problem what will concern us is that of the case in which a countable collection of machines M_j are all simultaneously trying to predict the sequence. Can we find a sequence which will be able to "baffle" all of them simultaneously? The answer is that if the countable class M_j is effectively defined then there exists an effectively defined sequence which will Baffle them all.*

* Indeed, we would obtain a satisfactory notion of randomness if we accept a sequence as random if and only if it baffles all effective prediction methods. But since the latter cannot be effectively enumerated, we reject this as an unrealistic definition.

Let us accept for the moment the definition that if $\alpha = 0$ for a particular machine and sequence, that machine has been fooled by that sequence. Now suppose that a <u>class $\{M_j\}$ of Turing machines has been defined effectively</u>, that is, there is given a "master" machine M which, given the "integer 'j' and a sequence S(0),S(1),..., S(t), computes what M_j would give for that same sequence. Suppose further (but informally, now) that this collection has been so chosen ad to include all known tests for statistical regularity, so that there is at least some basis for thinking of the machines as actually trying to make predictions.

It was shown by Minsky and Silver (Ref. 1) that under these conditions there always exists a computable sequence for which $\alpha = 0$ for every one of the machines M_j. The question of whether this remains true for machines which don't have to guess every time was unanswered in [1]. It is felt that this case is important, because it allows more flexible "statistician machines" which can wait for certain special events and only then guess. Thus, a machine might wait until there was a run of 1,000,000 consecutive 0's, add then guess that the next term will be 0. This would be a sound policy in the real world, for if the process is really random, one will do no worse than chance, while if the long run is due to a systematic defect in the experiment, that defect is liable to remain operative for some further time. The theorem was extended to this more general case by Levin, and the proof below is essentially that in his thesis [2].

This concept of guessing is now extended to allow the guessing machines to guess only some of the time. This seems to be a fairer

This means that we cannot use the simple score function described earlier, but must take the score relative to the number of actual guesses.

We now state a number of definitions. The sequence to be predicted is $S(t) = S(1), S(2), \ldots$ The sequence--or rather, its initial segments--are examined by a sequence of Turing machines M_j. It is required that each of these machines compute a <u>total</u> function over the set of all finite sequences. That is, given a finite sequence (of 0's and 1's) the machine will eventually halt, giving an output. The n^{th} guess of the machine M_j is written $M_j(n)$ and is the value M_j gives in response to the sequence $[S(1),\ldots,S(n-1)]$. This value must be either 0, 1, or Φ (which signifies "no guess").

We keep a running score $G_j(t)$ of the number of correct guesses minus the number of incorrect guesses. Let $H_j(t)$ be the number of times the machine has guessed rather than indicated Φ. Then we will say that if either

$$\lim_{t \to \infty} \frac{G_j(t)}{H_j(t)} = 0 \qquad \text{or if} \lim_{t \to \infty} H_j(t) < \infty,$$

the machine $M_j(t)$ has failed to predict the sequence $S(t)$. In the first case the machine guessed correctly no more than half the time. In the second case the machine stopped guessing and we cannot give it credit for being lucky on a finite number of guesses. In all other cases we consider the machine to have made a significant prediction of $S(t)$.

Returning to the idea of random nets, one might hope to define a sequence as random if it can pass a certain family of tests. By using the Turing Machines M_j, we restrict the tests to be effectively computable. We have required that the test values themselves be defined--that the machines terminate each computation.

One attempt at such a definition would simply include all possible guessing machines in the test family. A sequence whixh passes all such tests will surely appear random. Clearly these could exist no such computable sequence, but as shown by Church[3]*there must exist (non-computable) such sequences. However, this definition cannot be made effective, because the set of tests itself---the set of Turing machines which compute functions defined over the whole input range---cannot be effectively defined. Instead, we consider an arbitrary effectively enumerated sequence of tests---the sequence $\{M_j\}$ of machines defined above. Our theorem is

Given any effectively enumerated sequence of test machines M_j, there exists a computable sequence $S(t)$ which baffles all the M_j.

Since the sequence $\{M_j\}$ of machines is effectively enumerable, there exists a single machine that computes the same function as M_j when supplied with the integer j. That is, there is a machine U such that

$$M_j(t) = U(j,t).$$

To show how the sequence $S(t)$ is constructed, we begin with the special case in which there is only a finite number, n, on machines. Assume that S has been computed for all $t < t_0$. Thon We can compute $U(j,t)$ for all $1 \leq j \leq n$ and $1 \leq t \leq t_0$. For each t the guesses of the n machines form an n-vector $V[t] = [U(1,t),U(2,t),\ldots,U(n,t)]$. There are 3^n possible values for V[t]. To compute $S(t_0)$ we count the number of values of t for which $V[t]=V[t_0]$ and $t < t_0$. Assign the value 0 or 1 to $S(t_0)$ as this number is even or odd.

To illustrate, suppose there are only two machines, and they have just made their 10^{th} guess:

---

*But not for the optional guessing case. If we remove the condition in the theorem below, that the machines be effectively enumerable, we obtain Church's result, since the construction carries through (non-effectively) without this condition.

t	1	2	3	4	5	6	7	8	9	10
$M_1(t)$	1	0	Φ	Φ	0	0	1	Φ	0	1
$M_2(t)$	0	Φ	0	1	1	1	0	Φ	1	0
S(t)	0	0	0	0	0	1	1	0	0	?

The vector V [10] is (1,0). Since this identical vector has already occurred exactly twice before (V[10] = V[7] = V[1]), and two is even, the value of S(10) should be 0.

Now consider any vector type, for instance the vector type of V[1], V[7], and V[10]. The value of S(t) alternates strictly on the subsequence 1, 7, 10, ... If the correct-incorrect score $G_j(t)$ were kept only on this subsequence of t, then it could never become greater than 1. But there are at most 3^n such subsequences, corresponding to the 3^n vector types when there are n machines. Therefore:

$$2) \quad G_j(t) \leq 3^n$$

If for a given machine M_j, $\lim_{t\to\infty} H_j(t) = \infty$, this implies:

$$3) \quad \lim_{t\to} \frac{G_j(t)}{H_j(t)} = 0$$

Therefore M_j has failed to predict S(t).

To extend this technique to infinite sets of machines, we cannot consider infinite vectors, but we can increase the number of machines under consideration so as to include each of them eventually.

A value of S(t) obtained by applying the above recursive scheme to the first j machines will be called S'(j,t). The number j will be called the depth of the computation. In general, the depth tends

to increase with increasing t. By deciding on the correct depth for each t, we can compute S(t) by setting $S(t) = S'(j,t)$.

The rule for computing the depth is now given. Let t_0 be fixed and consider the sequence $S'(1,t_0)$, $S'(2,t_0)$, ... In computing $S'(j,t_0)$ a count was made on the number of earlier vectors that are identical to $V[t_0] = (S'(1,t_0),\ldots,S'(j,t_0))$. If this frequency count is greater than 3^{2j}, then the computation of $S'(j,t_0)$ will be called excessive. The correct depth (j) is now given as the smallest value of j for which $S'(j,t_0)$ is not excessive.

Having stated the algorithm, we must now prove that it works by establishing the following facts:

1. We shall consider a particular machine M_n and prove that it cannot predict the sequence S(t). If M_n makes only finitely many guesses, then there is nothing to prove. We assume that it makes infinitely many guesses for the rest of this proof.

2. For each guess $M_n(t)$, S(t) was made at a particular level j, and for a particular vector type $V[t] = (M_1(t),\ldots,M_j(t))$. At level j there are 3^j_j different vector types. At most 3^{2j} computations are made for each vector type before it becomes excessive. Therefore, at most 3^{3j} computations are made at each level.

This is to say, that if we rewrite S(t) to indicate the level of each computation by writing $S'(j_1,1)$, $S'(j_2,2),\ldots$, then each integer j_n will occur at most 3^{3j} times.

3. There are an infinite number of values of t for which $M_n(t) \neq \Phi$. Since only a finite number of them are made at any one level and since no level can be skipped, M_n guesses at all levels.

In fact, we can establish a minimum number of guesses atlevel j

by the following reasoning:

a. M_n guesses at least once at level j+1. This means that for some t, S'(j,t) is excessive for a vector $V[t] = [0_1(t), \ldots M_j(t)]$ such that $M_n(t) = 0$ or 1. Let T(j) be the least such t. Then the vector V[t] has occurred 3^{2j} times before time T(j). Note that the sequence $\{T(j)\}$, forms a strictly monotonic unbounded sequence.

b. After the vector V[t] occured $3^{2(j-1)}$ times the level (j-1) became excessive. Therefore $3^{2j} - 3^{2(j-1)} = \frac{8}{9}3^{2j}$ guesses, at least, have been made at level j.

4. We now wish to estimate $\frac{|G(t)|}{H(t)}$. Let T be the time such that for t>T, all guesses are made at level n or higher. (We are considering the score for machine M_n.) For all levels j<n the machine may do very well. But even if all its guesses are correct, there are only finitely many of them. To be precise, these can add at most T to the score G. [5].

Starting with level n, M_n guesses right only half the time on each vector type. At level j there are 3^j vector types and at least $\frac{8}{9}3^{2j}$ guesses are made. (See argument 3b.) Hence G(t) can change by at most 3^j at level j, while H(t) must increase by at least $\frac{8}{9}3^{2j}$.

5. Now for each t find the $j = J(t)$ such that $T(j) \leq t < T(j+1)$; that is, the largest j such that $T(j) \leq t$. We can then bound G(t) by

$$G(t) \leq T + \sum_{j=n}^{J(t)} 3^j,$$

for no guesses have been made at levels higher than J(t). We can set a lower bound on H(t) by the argument in (4) above:

$$H(t) \geq \frac{8}{9} \sum_{j=1}^{J(t)-1} 3^{2j}.$$

Now, by (3), as $t \to \infty$, $J(t) \to \infty$, so we have

$$0 \leq \lim_{t \to \infty} \left| \frac{\bar{G}(t)}{H(t)} \right| \leq \lim_{J(t) \to \infty} \frac{T + \sum_{j=n}^{J(t)} 3^j}{\frac{8}{9} \sum_{j=1}^{J(t)-1} 3^{2j}} = 0$$

Now if we replace all expressions of the form $M_j(t)$ by the corresponding $U(j,t)$, the construction becomes effective, $S(t)$ is a computable sequence, and the theorem is proved.

References:

[1] Minsky, M. L., "On some effective definitions of Random Sequences," Chap. III of Rept. 54-G-0023, M.I.T. Lincoln Laboratory, Lexington, Mass. June, 1960.

[2] Levin, M., B.S. Thesis in Mathematics, M.I.T., June 1961.

[3] Church, A., "On the concept of a Random Sequence," Bulletin of the American Mathematical Society, Vol. 46, (1940), pp. 130-135.

[4] Von Mises, R., Grundlagen der Wahrscheinlichkeitsrechnung, Mathematische Zeitschrift, Vol. 5, (1919), pp. 52-99.

[5] We are indebted to D. W. Loveland for observing and repairing the incorrect bound on G as it appears in [2].

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM-36

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)

☐ Other:_____

Document Information

Number of pages: 12 (16-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:

☐ Single-sided or

☒ Double-sided

Intended to be printed as :

☐ Single-sided or

☒ Double-sided

Print type:

☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: COPY OF MIMEOGRAPH

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): _____

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-12) TITLE PAGE & BLANK, 1-10	
(13-16) SCANCONTROL, TRGT'S (3)	

Scanning Agent Signoff:

Date Received: 11/30/95 Date Scanned: 12/12/95 Date Returned: 12/14/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**